Project 2: FDTD Modeling of a 3D PEC Cavity Resonator

Vicente Garcia

University of Colorado Denver

Dr. Stephen Gedney

ELEC 4333

April 3, 2025

TABLE OF CONTENTS

1. ABSTRACT

The Finite Difference Time Domain method is a technique widely used in computational electromagnetics because of the second order accuracy it provides to *Maxwell's equations* solutions. This paper presents a 3D-*FDTD* simulation over a rectangular *PEC* (Perfect Electric Conductor) cavity. The cavity is excited with an impressed current density in the positive z-direction with a broadband source. A differentiated Gaussian pulse is defined as the source signature. It is expected the software to demonstrate a 2nd order accuracy, as well as identifying and computing the 3 lowest resonant frequencies correspondingly to the cavity resonator. A FFT (*Fast Fourier Transformation*) facilitates the identification of peaks and resonant frequencies. Additionally, an error analysis will determine the *relative error* between computed and analytical resonance for TM (1,1,0) and (2,1,0) modes. Cases have been established for analysis where the number of cells for x, y, and z, is modified as well as the frequency sampling size for the FFT, *nfft*. The programming language chosen for software is *MATLAB*

Keywords: FDTD, relative error, cavity, differentiated gaussian pulse, nfft, maxwell equations, FFT (Fast Fourier Transform), PEC, MATLAB.

2. THEORY

The Finite Difference Time Domain method is one of the easiest methods to simulate radiation of electromagnetic waves over time and space. That is because it uses recursive updates to achieve 2nd order accuracy solutions to Maxwell's equations. For a 1D simulation, as done in project 1, Maxwell's Equations are treated along a transmission line resulting in convenient to use voltage $(\vec{V})$ and current $(\vec{I})$ terms. However, due to the complexity of a three-dimensional space, it results in convenient for this project to use fields $(\vec{E},\ \vec{H})$, and fluxes terms $(\vec{D},\vec{B})$ instead. Maxwell's Equations, in their differential form, consist of 4 equations listed below:

$$\text{Faraday's Law: } \frac{\partial \vec{B}}{\partial t} = -\nabla \text{ x } \vec{E} - \vec{M} \tag{1}$$

$$\text{Ampere's Law: } \frac{\partial \vec{D}}{\partial t} = \nabla \text{ x } \vec{H} - \vec{J} \tag{2}$$

$$\text{Gauss's Law: } \nabla \text{ x } \vec{D} = \rho \tag{3}$$

$$\text{Gauss's Law: } \nabla \text{ x } \vec{D} = \rho^*, \tag{4}$$

where $\vec{B}$ is the magnetic flux density (Wb/m^2), $\vec{D}$ is the electric flux density (C/m^2), also called as electric displacement, $\vec{E}$ is the electric field intensity (V/m), $\vec{H}$ is the magnetic field intensity (A/m), $\vec{J}$ is the electric current density (A/m^2), $\vec{M}$ is the magnetic current density (V/m^2), ρ is the electric charge density (C/m3), and ρ^* is the magnetic charge density (Wb/m^3). The flux densities and the field intensities are related through the equations listed below:

$$\vec{D} = \varepsilon\vec{E} = \varepsilon_0\varepsilon_r\vec{E}$$

(5)

$$\vec{B} = \mu\vec{H} = \mu_0\mu_r\vec{H} \,,$$

(6)

where ε_0 is the free-space permittivity (8.854 x 10^{-12} F/m), ε_r is the relative permittivity, μ_0 is the free-space permeability (4π x 10^{-7} H/m), and μ_r is the relative permeability. It is important to note that these fields and fluxes are subject to boundary conditions. These conditions depend on the material and source properties; for instance, at this project a PEC is used to represent the surface of a lossless metallic conductor, where a special treatment of the boundary conditions at surface *S* is required. These conditions are:

$$\hat{n} \times \vec{E}_1|_{S\,PEC} = 0, \qquad \hat{n} \times \vec{H}_1|_{S\,PEC} = \vec{J}_S$$

(7)

$$\hat{n} \cdot \vec{D}_1|_{S\,PEC} = \rho_S, \qquad \hat{n} \cdot \vec{B}_1|_{S\,PEC} = 0.$$

(8)

Moving forward, a Yee's Algorithm is studied for effectively implementing the FDTD method over a three-dimensional space. Kane S. Yee based his algorithm on discretizing a physical space over the x, y and z domain with a staggered cubical grid. A so called "Primary Grid" will represent the projection of Faraday' Law as seen in Figure 1. In the same way, a "Secondary Grid" will represent Ampere's Law staggered half a cell in all directions. The secondary grid edges connect the center of the primary grid cell, as illustrated in Figure 2.

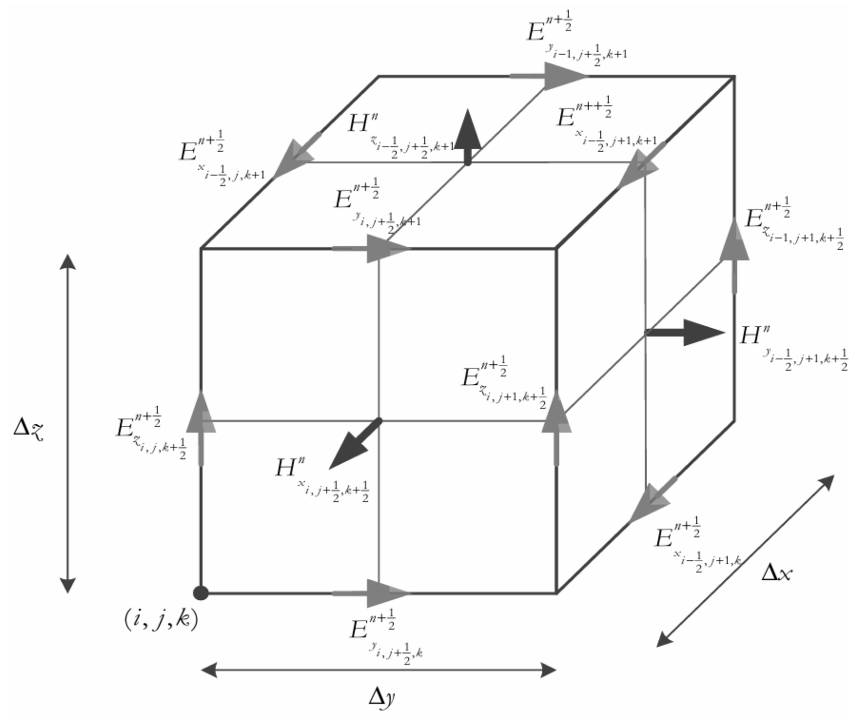

$E^{n+\frac{1}{2}}_{y_{i-1,j+\frac{1}{2},k+1}}$
$H^{n}_{z_{i-\frac{1}{2},j+\frac{1}{2},k+1}}$
$E^{n++\frac{1}{2}}_{x_{i-\frac{1}{2},j+1,k+1}}$
$E^{n+\frac{1}{2}}_{x_{i-\frac{1}{2},j,k+1}}$
$E^{n+\frac{1}{2}}_{y_{i,j+\frac{1}{2},k+1}}$
$E^{n+\frac{1}{2}}_{z_{i-1,j+1,k+\frac{1}{2}}}$
$H^{n}_{y_{i-\frac{1}{2},j+1,k+\frac{1}{2}}}$
$E^{n+\frac{1}{2}}_{z_{i,j+1,k+\frac{1}{2}}}$
Δz
$E^{n+\frac{1}{2}}_{z_{i,j,k+\frac{1}{2}}}$
$H^{n}_{x_{i,j+\frac{1}{2},k+\frac{1}{2}}}$
$E^{n+\frac{1}{2}}_{x_{i-\frac{1}{2},j+1,k}}$
Δx
(i,j,k)
$E^{n+\frac{1}{2}}_{y_{i,j+\frac{1}{2},k}}$
Δy

Figure 1: Yee’s Primary Grid cell.

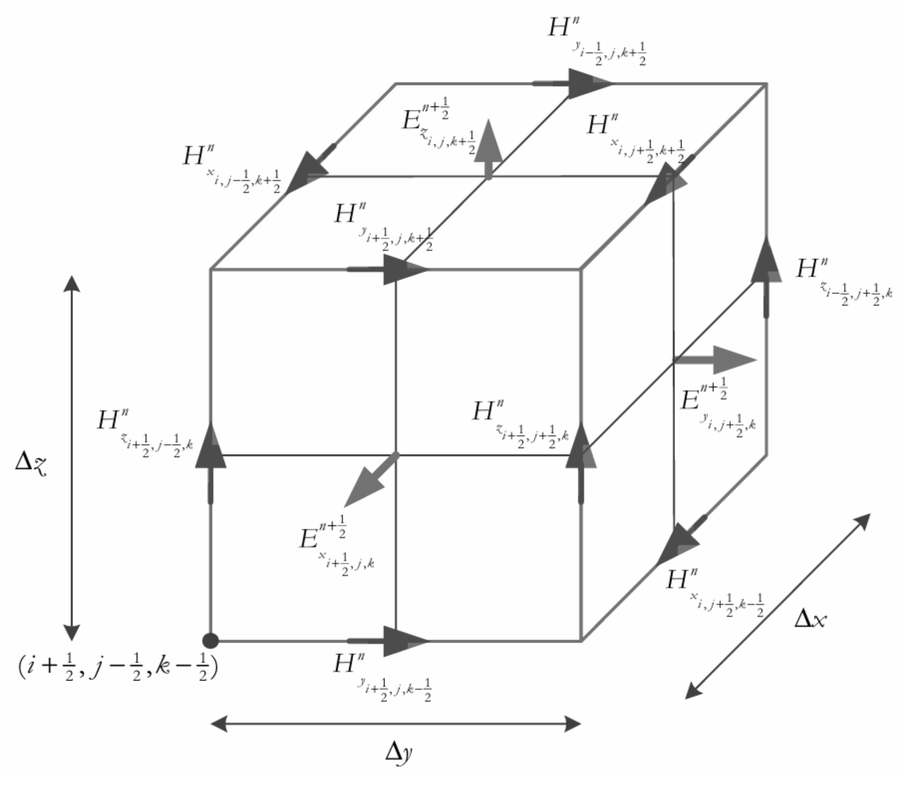

$H^{n}_{y_{i-\frac{1}{2},j,k+\frac{1}{2}}}$
$E^{n+\frac{1}{2}}_{z_{i,j,k+\frac{1}{2}}}$
$H^{n}_{x_{i,j+\frac{1}{2},k+\frac{1}{2}}}$
$H^{n}_{x_{i,j-\frac{1}{2},k+\frac{1}{2}}}$
$H^{n}_{y_{i+\frac{1}{2},j,k+\frac{1}{2}}}$
$H^{n}_{z_{i-\frac{1}{2},j+\frac{1}{2},k}}$
$E^{n+\frac{1}{2}}_{y_{i,j+\frac{1}{2},k}}$
$H^{n}_{z_{i+\frac{1}{2},j-\frac{1}{2},k}}$
$H^{n}_{z_{i+\frac{1}{2},j+\frac{1}{2},k}}$
Δz
$E^{n+\frac{1}{2}}_{x_{i+\frac{1}{2},j,k}}$
$H^{n}_{x_{i,j+\frac{1}{2},k-\frac{1}{2}}}$
Δx
$(i+\frac{1}{2},j-\frac{1}{2},k-\frac{1}{2})$
$H^{n}_{y_{i+\frac{1}{2},j,k-\frac{1}{2}}}$
Δy

Figure 2: Yee’s Secondary Grid cell.

Six recursive update equations, 3 for the electric field, and 3 for the magnetic field describe the response of a source excitation in a rectangular PEC cavity. As seen from equations (9)-

$$E_{z_{i,j,k+\frac{1}{2}}}^{n+\frac{1}{2}} = E_{z_{i,j,k+\frac{1}{2}}}^{n-\frac{1}{2}} + \frac{\Delta t}{\varepsilon}\left(\frac{H_{y_{i+\frac{1}{2},j,k+\frac{1}{2}}}^{n} - H_{y_{i-\frac{1}{2},j,k+\frac{1}{2}}}^{n}}{\Delta x}\right) - \; H_{x_{i,j+\frac{1}{2},k+\frac{1}{2}}}^{n} -$$

$$H_x \square i,j - \frac{1}{2}, k + \frac{1}{2} \square n \square \square \Delta y \square \square - J_{z_{i,j,k+\frac{1}{2}}}^{n} \square .$$

(*14*) subscripts such as *i, j, k* are defined for space in the x, y and z direction respectively, and *n* for time. The recursive update equations for *H*, and *E* fields are the following:

$$H_x{}_{i,j+\frac{1}{2},k+\frac{1}{2}}^{n+1} = H_x{}_{i,j+\frac{1}{2},k+\frac{1}{2}}^{n} + \frac{\Delta t}{\mu}\left[\left(\frac{E_y{}_{i,j+\frac{1}{2},k+1}^{n+\frac{1}{2}} - E_y{}_{i,j+\frac{1}{2},k}^{n+\frac{1}{2}}}{\Delta z}\right) - \left(\frac{E_z{}_{i,j+1,k+\frac{1}{2}}^{n+\frac{1}{2}} - E_z{}_{i,j,k+\frac{1}{2}}^{n+\frac{1}{2}}}{\Delta y}\right) - M_x{}_{i,j+\frac{1}{2},k+\frac{1}{2}}^{n+\frac{1}{2}}\right] \tag{9}$$

$$H_y{}_{i+\frac{1}{2},j,k+\frac{1}{2}}^{n+1} = H_y{}_{i+\frac{1}{2},j,k+\frac{1}{2}}^{n} + \frac{\Delta t}{\mu}\left[\left(\frac{E_z{}_{i+1,j,k+\frac{1}{2}}^{n+\frac{1}{2}} - E_z{}_{i,j,k+\frac{1}{2}}^{n+\frac{1}{2}}}{\Delta x}\right) - \left(\frac{E_x{}_{i+\frac{1}{2},j,k+1}^{n+\frac{1}{2}} - E_x{}_{i+\frac{1}{2},j,k}^{n+\frac{1}{2}}}{\Delta z}\right) - M_y{}_{i+\frac{1}{2},j,k+\frac{1}{2}}^{n+\frac{1}{2}}\right] \tag{10}$$

$$H_{z}{}_{i+\frac{1}{2},j+\frac{1}{2},k}^{n+1} = H_{z}{}_{i+\frac{1}{2},j+\frac{1}{2},k}^{n} + \frac{\Delta t}{\mu}\left[\left(\frac{E_{x}{}_{i+\frac{1}{2},j+1,k}^{n+\frac{1}{2}} - E_{x}{}_{i+\frac{1}{2},j,k}^{n+\frac{1}{2}}}{\Delta y}\right) - \left(\frac{E_{z}{}_{i+1,j+\frac{1}{2},k}^{n+\frac{1}{2}} - E_{z}{}_{i,j+\frac{1}{2},k}^{n+\frac{1}{2}}}{\Delta x}\right) - M_{z}{}_{i+\frac{1}{2},j+\frac{1}{2},k}^{n+\frac{1}{2}}\right] \tag{11}$$

$$E_{x}{}_{i+\frac{1}{2},j,k}^{n+1/2} = E_{x}{}_{i+\frac{1}{2},j,k}^{n-1/2} + \frac{\Delta t}{\varepsilon}\left[\left(\frac{H_{z}{}_{i+\frac{1}{2},j+\frac{1}{2},k}^{n} - H_{z}{}_{i+\frac{1}{2},j-\frac{1}{2},k}^{n}}{\Delta y}\right) - \left(\frac{H_{y}{}_{i+\frac{1}{2},j,k+\frac{1}{2}}^{n} - H_{y}{}_{i+\frac{1}{2},j,k-\frac{1}{2}}^{n}}{\Delta z}\right) - J_{x}{}_{i+\frac{1}{2},j,k}^{n+\frac{1}{2}}\right], \tag{12}$$

where *J* represents the current density impressed as source, and *M* represents the magnetic contribution to that current density. Similarly for E_y, and E_z:

$$E_y{}^{n+\frac{1}{2}}_{i,j+\frac{1}{2},k} = E_y{}^{n-\frac{1}{2}}_{i,j+\frac{1}{2},k} + \frac{\Delta t}{\varepsilon}\left[\left(\frac{H_x{}^{n}_{i,j+\frac{1}{1},k+\frac{1}{2}} - H_x{}^{n}_{i,j+\frac{1}{2},k-\frac{1}{2}}}{\Delta z}\right) - \left(\frac{H_z{}^{n}_{i+\frac{1}{2},j+\frac{1}{2},k} - H_z{}^{n}_{i-\frac{1}{2},j+\frac{1}{2},k}}{\Delta x}\right) - J_y{}^{n+\frac{1}{2}}_{i,j+\frac{1}{2},k}\right]$$

(13)

$$E_z{}^{n+\frac{1}{2}}_{i,j,k+\frac{1}{2}} = E_z{}^{n-\frac{1}{2}}_{i,j,k+\frac{1}{2}} + \frac{\Delta t}{\varepsilon}\left[\left(\frac{H_y{}^{n}_{i+\frac{1}{2},j,k+\frac{1}{2}} - H_y{}^{n}_{i-\frac{1}{2},j,k+\frac{1}{2}}}{\Delta x}\right) - \left(\frac{H_x{}^{n}_{i,j+\frac{1}{2},k+\frac{1}{2}} - H_x{}^{n}_{i,j-\frac{1}{2},k+\frac{1}{2}}}{\Delta y}\right) - J_z{}^{n}_{i,j,k+\frac{1}{2}}\right].$$

(14)

It is important to emphasize that equations (9)-$Ez⍰ i,j,k+\frac{1}{2} ⍰ n+\frac{1}{2} ⍰ = E_z{}^{n-\frac{1}{2}}_{i,j,k+\frac{1}{2}} + \frac{\Delta t}{\varepsilon}\left[\left(\frac{H_y{}^{n}_{i+\frac{1}{2},j,k+\frac{1}{2}} - H_y{}^{n}_{i-\frac{1}{2},j,k+\frac{1}{2}}}{\Delta x}\right) - \left(\frac{H_x{}^{n}_{i,j+\frac{1}{2},k+\frac{1}{2}} - H_x{}^{n}_{i,j-\frac{1}{2},k+\frac{1}{2}}}{\Delta y}\right) - J_z{}^{n}_{i,j,k+\frac{1}{2}}\right].$

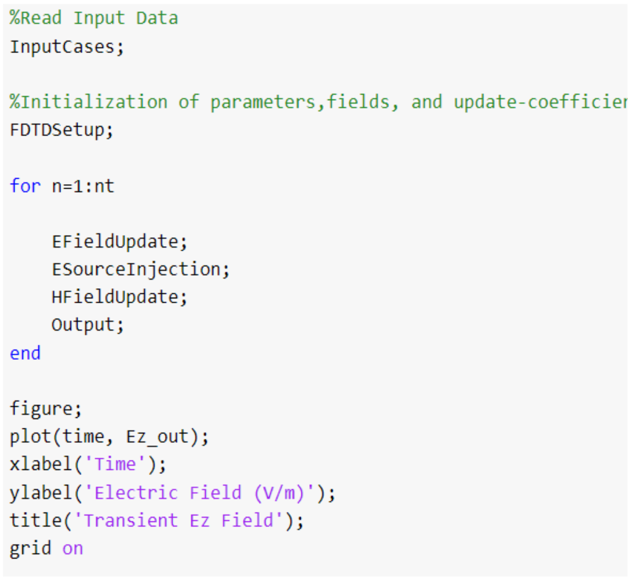
(*14*) provide an explicit recursive update of the fields in a linear, isotropic, and lossless medium. Added to that, the time step for these update equations must satisfy the stability criteria, otherwise, the energy in the system will grow unboundedly. The Courant-Fredisch-Lewy (CFL) stability Limit is the following:

$$\Delta t < \frac{1}{c} * \frac{1}{\sqrt{\frac{1}{\Delta x^2}+\frac{1}{\Delta y^2}+\frac{1}{\Delta z^2}}} * CFLN\,, \tag{15}$$

where c is the wave speed, $c = \frac{1}{\sqrt{\mu_0 * \varepsilon_0}}$.

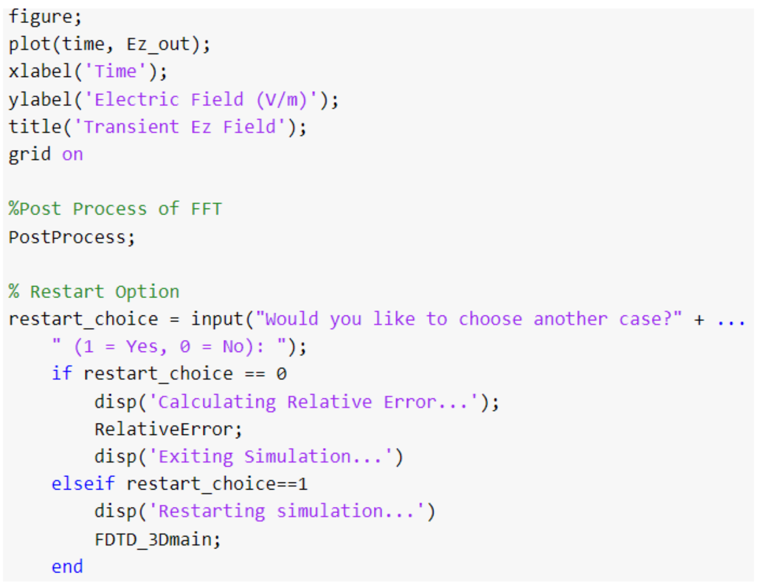
3. CODE DESCRIPTION

A top-level code design has been developed to compute the recursive update equations for specific input cases. The code follows the flow chart design illustrated in Figure 5. The main script contains the input parameters, the *E* and *H* field updates (including its source signature), the output control, and lastly the post processing parameters as observed in Figure 3 and 4. A relative error script is added after post-process to calculate the relative error between measured (computed), and theoretical (analytical) frequencies for the TM(1,1,0) and TM(2,1,0) modes. These blocks have been defined as scripts instead of functions due to simplicity.



```
%Read Input Data
InputCases;

%Initialization of parameters,fields, and update-coefficier
FDTDSetup;

for n=1:nt

    EFieldUpdate;
    ESourceInjection;
    HFieldUpdate;
    Output;
end

figure;
plot(time, Ez_out);
xlabel('Time');
ylabel('Electric Field (V/m)');
title('Transient Ez Field');
grid on
```

Figure 3: Main script.



```
figure;
plot(time, Ez_out);
xlabel('Time');
ylabel('Electric Field (V/m)');
title('Transient Ez Field');
grid on

%Post Process of FFT
PostProcess;

% Restart Option
restart_choice = input("Would you like to choose another case?" + ...
    " (1 = Yes, 0 = No): ");
    if restart_choice == 0
        disp('Calculating Relative Error...');
        RelativeError;
        disp('Exiting Simulation...')
    elseif restart_choice==1
        disp('Restarting simulation...')
        FDTD_3Dmain;
    end
```

Figure 4: Main script ctd.

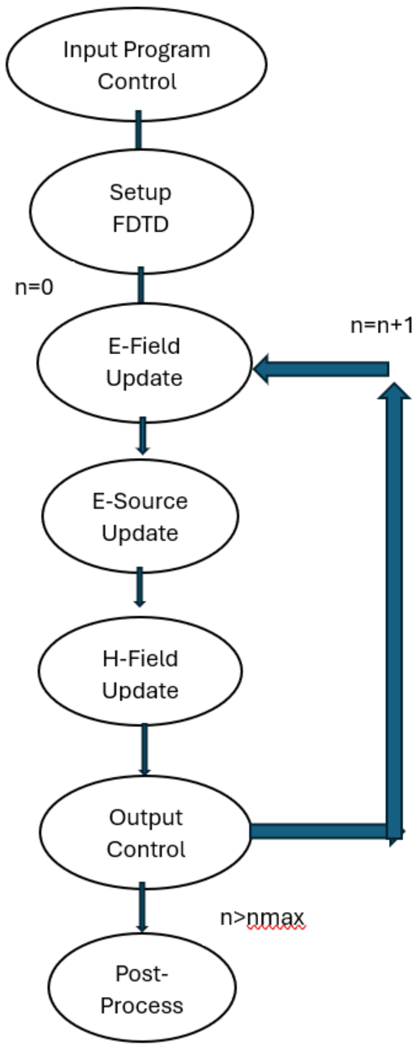
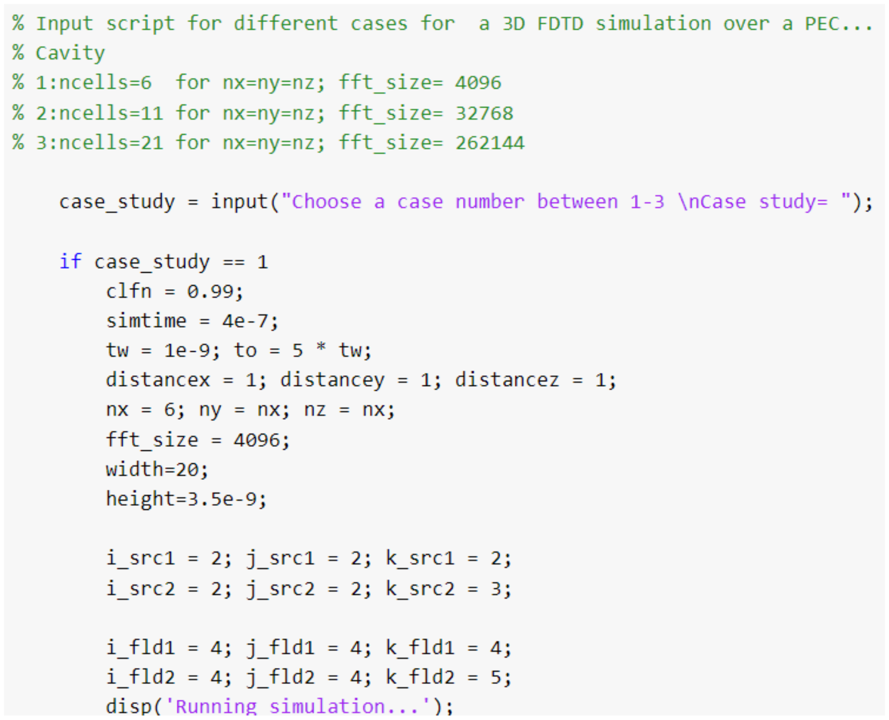


Figure 5: Flow chart design for code description.

3.1. INPUT PROGRAM CONTROL

To begin with, the input program control includes 3 input cases that relate to each case study provided. These are selected by the user under the "*InputCase*" script. As evidenced in Figure 6, the cavity dimensions, as well as the simulation time, CFL number, source signature (differentiated Gaussian pulse), number of x, y and z cells, and the output control parameters are defined. The PEC cavity dimensions remain fixed for all cases with a volume of $1m^3$. The source signature time width, $tw = 1ns,\ t_0 = tw * 5$, and the CFL number, *clfn*=0.99 also remains fixed.



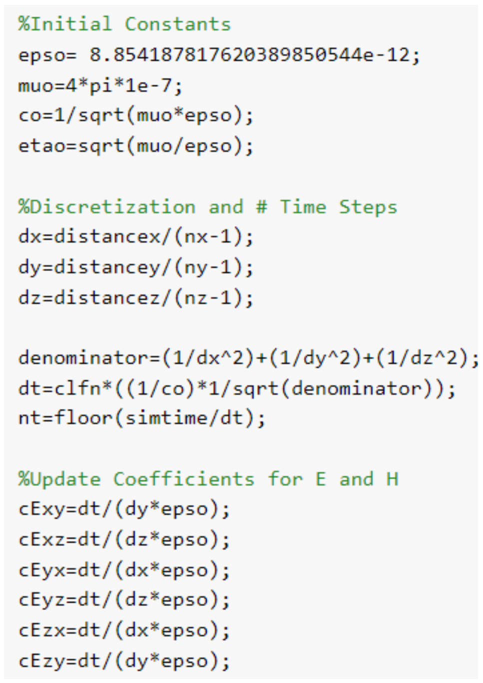
```
% Input script for different cases for  a 3D FDTD simulation over a PEC...
% Cavity
% 1:ncells=6  for nx=ny=nz; fft_size= 4096
% 2:ncells=11 for nx=ny=nz; fft_size= 32768
% 3:ncells=21 for nx=ny=nz; fft_size= 262144

    case_study = input("Choose a case number between 1-3 \nCase study= ");

    if case_study == 1
        clfn = 0.99;
        simtime = 4e-7;
        tw = 1e-9; to = 5 * tw;
        distancex = 1; distancey = 1; distancez = 1;
        nx = 6; ny = nx; nz = nx;
        fft_size = 4096;
        width=20;
        height=3.5e-9;

        i_src1 = 2; j_src1 = 2; k_src1 = 2;
        i_src2 = 2; j_src2 = 2; k_src2 = 3;

        i_fld1 = 4; j_fld1 = 4; k_fld1 = 4;
        i_fld2 = 4; j_fld2 = 4; k_fld2 = 5;
        disp('Running simulation...');
```

Figure 6: Input Case script.

Namely, case 1 has a number of cells, *nx, ny, nz* equal to 6, a sampling frequency, *fft_size,* equal to 2^{12}, a *width* of 20, a *height* of 3.5e-9, a source injection between nodes (2,2,2) and (2,2,3), and an output probe between nodes (4,4,4) and (4,4,5). In the same way, case 2 has a number of cells, *nx, ny, nz* equal to 11, a sampling frequency, *fft_size,* equal to 2^{15}, a *width* of 80, a *height* of 1.5e-9, a source injection between nodes (4,4,4) and (4,4,6), and an output probe between nodes (8,8,8) and (8,8,9). Case 3 has a number of cells, *nx, ny, nz* equal to 21, a sampling frequency, *fft_size,* equal to 2^{18}, a *width* of 320, a *height* of 3.9e-10, a source injection between nodes (8,8,8) and (8,8,12), and an output probe between nodes (16,16,16) and (16,16,17). It is valuable to remark that, the *fft_size, height* and *width* are parameters that facilitate the identification of resonant frequencies after the Fourier transformation. Figure 6 only contains input case 1 for the sake of spacing, nevertheless case 2 and 3 are defined in the same way with the variables mentioned previously.

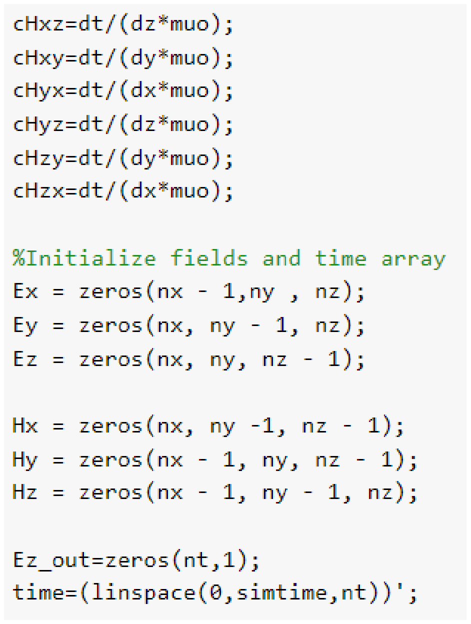
3.2. FDTD SETUP

Moving forward with the "*FDTD Setup*" block, initial constants are pre-established for the free-space permittivity and permeability, ε_0, μ_0, and the wave speed, c_0 as seen in Figure 7. The spaces discretization, *dx, dy,* and *dz* are computed given the number of cells, and the cavity dimensions, 1m^3. On the other hand, the time discretization is computed based on *dx, dy,* and *dz* values, as well as c_0, and the *clf* number. This will ease the calculation of time steps, *nt*, for the *E* and *H* field updates. Figures 7 and 8 exhibits the update coefficients for both fields; these are calculated given the time and spaces discretization (*dx, dy, dz*), with its initial constants ε_0, μ_0. A matrix of zeros is defined for *E* and *H* in the three-dimensional domain along with the time, and the output *E* field values, which are useful for storing them along the sequence *n=1* to *nt*



```
%Initial Constants
epso= 8.854187817620389850544e-12;
muo=4*pi*1e-7;
co=1/sqrt(muo*epso);
etao=sqrt(muo/epso);

%Discretization and # Time Steps
dx=distancex/(nx-1);
dy=distancey/(ny-1);
dz=distancez/(nz-1);

denominator=(1/dx^2)+(1/dy^2)+(1/dz^2);
dt=clfn*((1/co)*1/sqrt(denominator));
nt=floor(simtime/dt);

%Update Coefficients for E and H
cExy=dt/(dy*epso);
cExz=dt/(dz*epso);
cEyx=dt/(dx*epso);
cEyz=dt/(dz*epso);
cEzx=dt/(dx*epso);
cEzy=dt/(dy*epso);
```

Figure 7: FDTD Setup script



```
cHxz=dt/(dz*muo);
cHxy=dt/(dy*muo);
cHyx=dt/(dx*muo);
cHyz=dt/(dz*muo);
cHzy=dt/(dy*muo);
cHzx=dt/(dx*muo);

%Initialize fields and time array
Ex = zeros(nx - 1,ny , nz);
Ey = zeros(nx, ny - 1, nz);
Ez = zeros(nx, ny, nz - 1);

Hx = zeros(nx, ny -1, nz - 1);
Hy = zeros(nx - 1, ny, nz - 1);
Hz = zeros(nx - 1, ny - 1, nz);

Ez_out=zeros(nt,1);
time=(linspace(0,simtime,nt))';
```

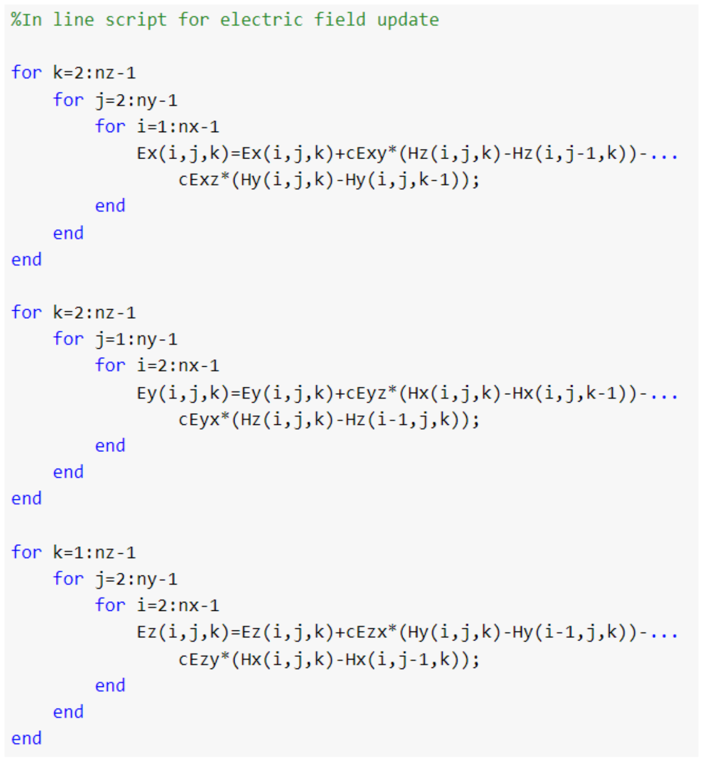
Figure 8: FDTD Setup script ctd.

3.3. ELECTRIC FIELD UPDATE

Furthermore, the "*Electric field update"* script runs the recursive updates equations along the PEC cavity as noted in Figure 9. A "*for n=1:nt"* loop is used for defining all the time steps over the simulation. It's important to note that Equations (*9*)-(*14*) need to be slightly modified for readability with MATLAB syntax as evidenced in Equation (16). This integer-based array indexing applies for all update equations in the x, y and z direction. The Electric field update equation in the z direction is written as:

$$E_z(i,j,k) = E_z(i,j,k) + \frac{\Delta t}{\varepsilon}\left[\frac{1}{\Delta x}\Big(H_y(i,j,k) - H_y(i-1,j,k)\Big) - \frac{1}{\Delta y}\big(H_x(i,j,k) - H_x(i,j-1,k)\big)\right], \tag{16}$$

It is important to remark that the outer boundaries are not valid for the electric field updates equations. This is because the update on the outer boundaries relies on the magnetic field outside the cavity dimension domain. Namely for the E_z field, the array indices go from *i=2: nx−1, j=2: ny−1,* and *k=1:nz−1*. Similarly, for the E_y field, the array indices go from *i=2: nx−1, j=1: ny−1,* and *k =2:nz−1*, and for the E_x field, from *i=1:nx −1,j=2:ny −1,* and *k =2 :nz−1.*



```
%In line script for electric field update

for k=2:nz-1
    for j=2:ny-1
        for i=1:nx-1
            Ex(i,j,k)=Ex(i,j,k)+cExy*(Hz(i,j,k)-Hz(i,j-1,k))-...
                cExz*(Hy(i,j,k)-Hy(i,j,k-1));
        end
    end
end

for k=2:nz-1
    for j=1:ny-1
        for i=2:nx-1
            Ey(i,j,k)=Ey(i,j,k)+cEyz*(Hx(i,j,k)-Hx(i,j,k-1))-...
                cEyx*(Hz(i,j,k)-Hz(i-1,j,k));
        end
    end
end

for k=1:nz-1
    for j=2:ny-1
        for i=2:nx-1
            Ez(i,j,k)=Ez(i,j,k)+cEzx*(Hy(i,j,k)-Hy(i-1,j,k))-...
                cEzy*(Hx(i,j,k)-Hx(i,j-1,k));
        end
    end
end
```

Figure 9: Electric field update script.

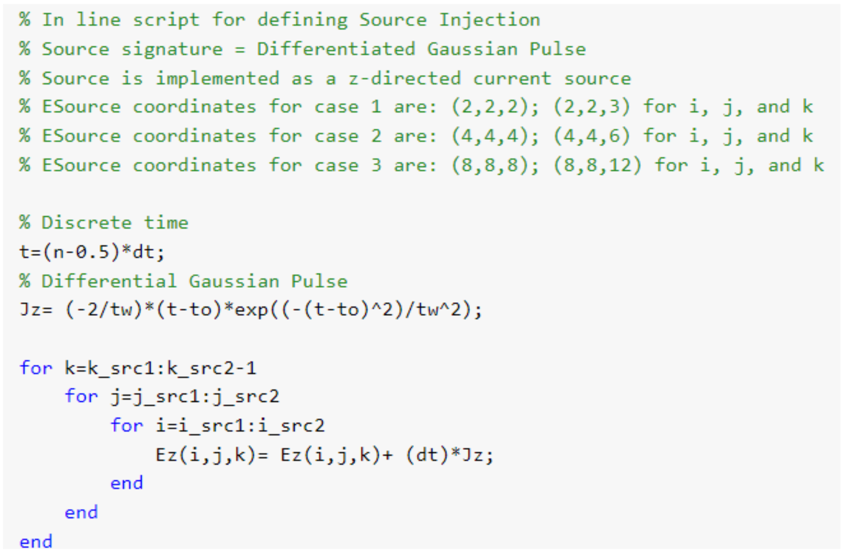
3.4. ELECTRIC FIELD SOURCE INJECTION

Moreover, a differentiated gaussian pulse is set as a current density source, J_z with its time width signature $tw = 1ns,\ t_0 = tw * 5$. The current density source in the z direction is computed as:

$$J_z = \frac{-2}{t_w}(t - t_o)\exp\left(-\frac{(t-t_o)^2}{t_w^2}\right). \tag{17}$$

The injection coordinates are provided accordingly with each case study as seen in Figure 10. Case 1 source is injected in the z-edge between nodes (2,2,2), and (2,2,3). Case 2 source is injected in the z-edge between nodes (4,4,4) and (4,4,6), and case 3 is injected in the z-edge between nodes (8,8,8) and (8,8,12). Given these parameters, the Electric Field source injection is expressed as:

$$E_z(i,j,k) = E_z(i,j,k) + dt * J_z. \tag{18}$$



```
% In line script for defining Source Injection
% Source signature = Differentiated Gaussian Pulse
% Source is implemented as a z-directed current source
% ESource coordinates for case 1 are: (2,2,2); (2,2,3) for i, j, and k
% ESource coordinates for case 2 are: (4,4,4); (4,4,6) for i, j, and k
% ESource coordinates for case 3 are: (8,8,8); (8,8,12) for i, j, and k

% Discrete time
t=(n-0.5)*dt;
% Differential Gaussian Pulse
Jz= (-2/tw)*(t-to)*exp((-(t-to)^2)/tw^2);

for k=k_src1:k_src2-1
    for j=j_src1:j_src2
        for i=i_src1:i_src2
            Ez(i,j,k)= Ez(i,j,k)+ (dt)*Jz;
        end
    end
end
```

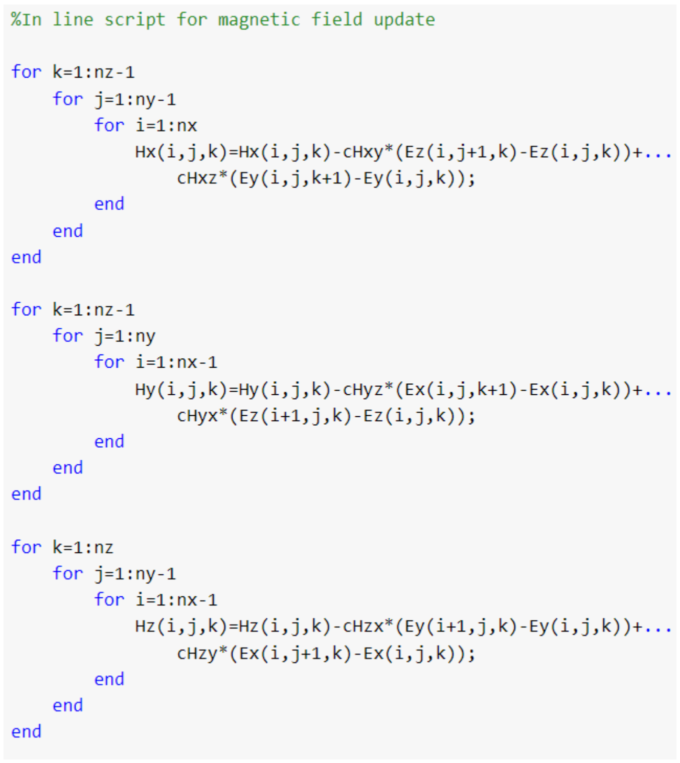
Figure 10: Electric Field source injection script.

3.5. MAGNETIC FIELD UPDATE

Likewise, section 3.3, the magnetic field equations are also updated along the loop *n=1: nt*, as shown in Figure 11. Again, an integer-based array indexing applies for all update equations in the x, y and z direction. The Magnetic field update equation in the z direction is written as:

$$H_z(i,j,k) = H_z(i,j,k) + \frac{\Delta t}{\mu}\left[\frac{1}{\Delta y}\left(E_x(i,j+1,k) - E_x(i,j,k)\right) - \frac{1}{\Delta x}\left(E_y(i+1,j,k) - E_y(i,j,k)\right)\right], \tag{19}$$

where the array indices go from, *i=1: nx−1, j=1: ny-1,* and *k =1: nz*. Similarly, for the H_y field, the array indices go from *i=1: nx−1, j=1: ny,* and *k=1: nz−1*, and for the H_x field, from *i =1: nx, j =1: ny −1,* and *k =1: nz−1.*



```
%In line script for magnetic field update

for k=1:nz-1
    for j=1:ny-1
        for i=1:nx
            Hx(i,j,k)=Hx(i,j,k)-cHxy*(Ez(i,j+1,k)-Ez(i,j,k))+...
                cHxz*(Ey(i,j,k+1)-Ey(i,j,k));
        end
    end
end

for k=1:nz-1
    for j=1:ny
        for i=1:nx-1
            Hy(i,j,k)=Hy(i,j,k)-cHyz*(Ex(i,j,k+1)-Ex(i,j,k))+...
                cHyx*(Ez(i+1,j,k)-Ez(i,j,k));
        end
    end
end

for k=1:nz
    for j=1:ny-1
        for i=1:nx-1
            Hz(i,j,k)=Hz(i,j,k)-cHzx*(Ey(i+1,j,k)-Ey(i,j,k))+...
                cHzy*(Ex(i,j+1,k)-Ex(i,j,k));
        end
    end
end
```

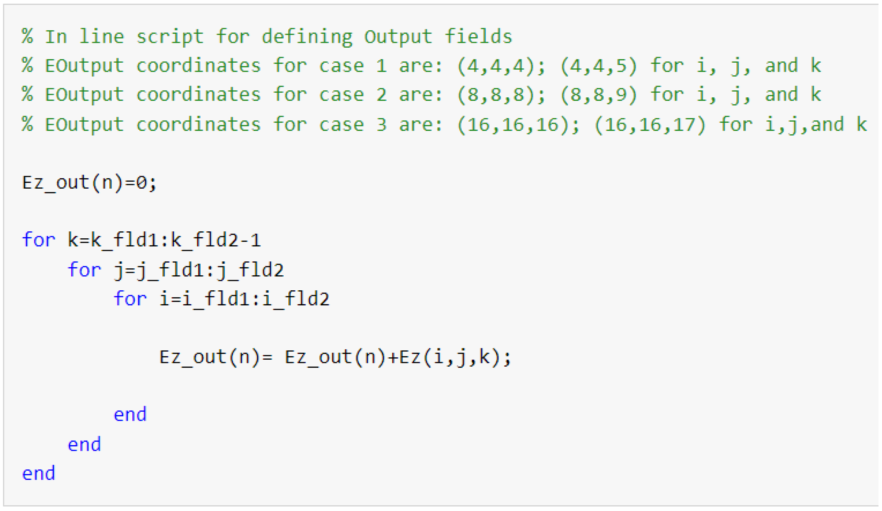
Figure 11: Magnetic field update script.

3.6. ELECTRIC FIELD OUTPUT

As part of the electric field output script, the update equations are computed similarly as the source injection in section 3.4, although instead, output probe coordinates are provided for each case study. As evidenced in Figure 12, case 1 output probe is set in the z-edge between nodes (4,4,4), and (4,4,5). Case 2 output probe is set in the z-edge between nodes (8,8,8) and (8,8,9), and case 3 output probe is set in the z-edge between nodes (16,16,16) and (16,16,17). Given these parameters, the Electric field output is expressed as:

$$E_z out(n) = E_z out(n) + E_z(i, j, k), \tag{20}$$

where $E_z out(n) = 0$ as initial condition seen in Figure 12.



```
% In line script for defining Output fields
% EOutput coordinates for case 1 are: (4,4,4); (4,4,5) for i, j, and k
% EOutput coordinates for case 2 are: (8,8,8); (8,8,9) for i, j, and k
% EOutput coordinates for case 3 are: (16,16,16); (16,16,17) for i,j,and k

Ez_out(n)=0;

for k=k_fld1:k_fld2-1
    for j=j_fld1:j_fld2
        for i=i_fld1:i_fld2

            Ez_out(n)= Ez_out(n)+Ez(i,j,k);

        end
    end
end
```

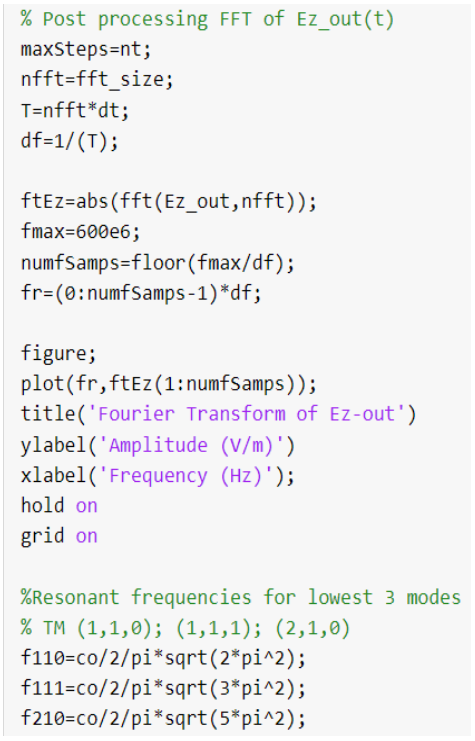
Figure 12: Electric Field output script.

3.7. POST-PROCESSING

At the post-processing block, a Fast Fourier Transformation is performed to facilitate the identification of resonant frequencies. A specific sampling size, "*nfft*" is defined at all cases for signal resolution. Initially, the frequency sampling should be greater than twice the period, meaning that resolution can be improved by decreasing the discrete samples of frequencies. Case 1 has a *nfft* value of 2^{12}, while case 2 has a *nfft* value of 2^{15}, and case 3 a *nfft* value of 2^{18}. In other words, the frequency discretization, *df,* decreases by a factor of 4 from the previous case. Added to that, it's evident that the simulation contains errors due to dispersion; therefore, to visualize these differences, the exact resonant frequencies for the 3 lowest modes are stored in the *fex* array as observed in Figure 13. These values are 2.119853e+08, 2.596279e+08, and 3.351781e+08. The *fex* array is calculated from the following equation:

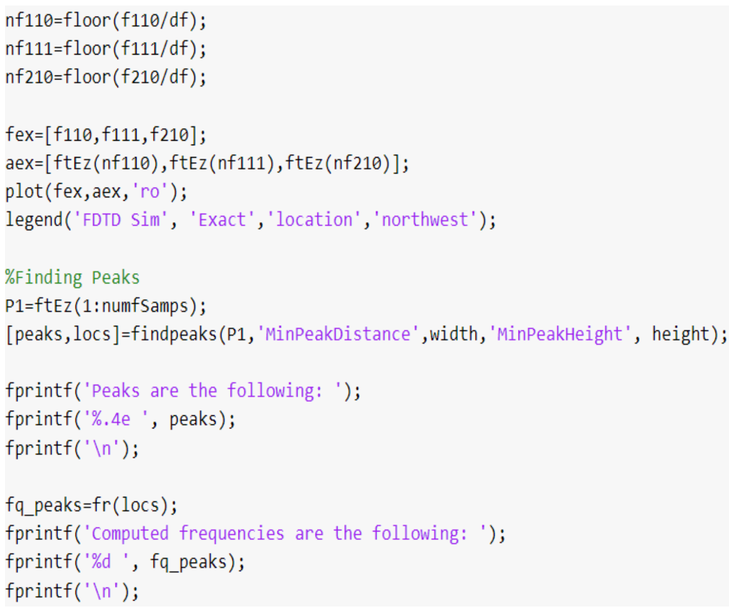
$$f_{m,n,l} = \frac{c}{2\pi}\sqrt{\left(\frac{m\pi}{a}\right)^2 + \left(\frac{n\pi}{b}\right)^2 + \left(\frac{l\pi}{d}\right)^2}, \tag{21}$$

where *m, n*, and *l* represent the TM mode, and *a, b*, and *d* represent the dimension for *x, y,* and *z* respectively.



```
% Post processing FFT of Ez_out(t)
maxSteps=nt;
nfft=fft_size;
T=nfft*dt;
df=1/(T);

ftEz=abs(fft(Ez_out,nfft));
fmax=600e6;
numfSamps=floor(fmax/df);
fr=(0:numfSamps-1)*df;

figure;
plot(fr,ftEz(1:numfSamps));
title('Fourier Transform of Ez-out')
ylabel('Amplitude (V/m)')
xlabel('Frequency (Hz)');
hold on
grid on

%Resonant frequencies for lowest 3 modes
% TM (1,1,0); (1,1,1); (2,1,0)
f110=co/2/pi*sqrt(2*pi^2);
f111=co/2/pi*sqrt(3*pi^2);
f210=co/2/pi*sqrt(5*pi^2);
```

Figure 13: Post-Processing script for FFT.

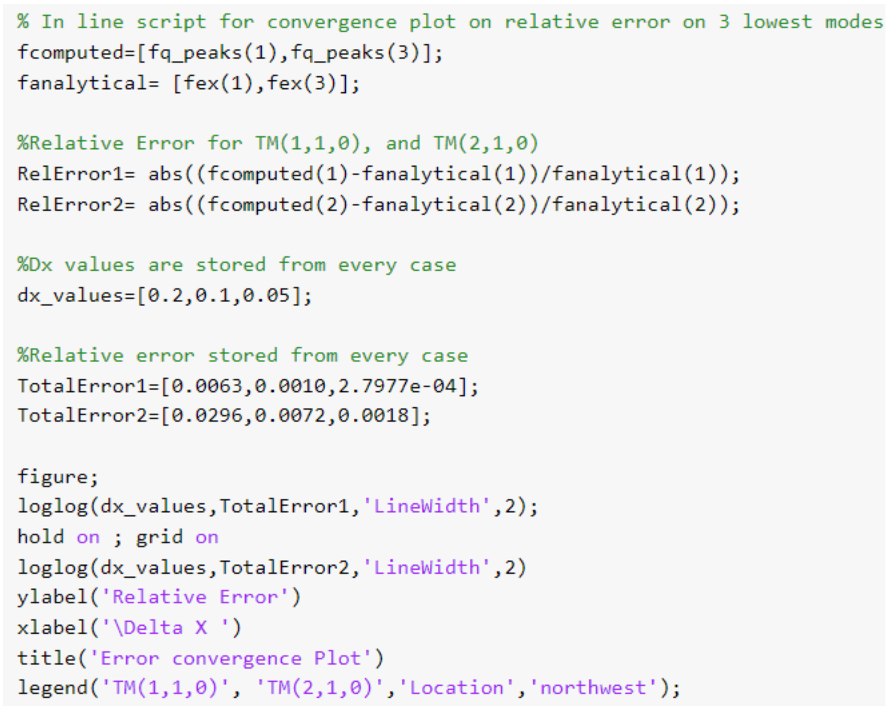


```
nf110=floor(f110/df);
nf111=floor(f111/df);
nf210=floor(f210/df);

fex=[f110,f111,f210];
aex=[ftEz(nf110),ftEz(nf111),ftEz(nf210)];
plot(fex,aex,'ro');
legend('FDTD Sim', 'Exact','location','northwest');

%Finding Peaks
P1=ftEz(1:numfSamps);
[peaks,locs]=findpeaks(P1,'MinPeakDistance',width,'MinPeakHeight', height);

fprintf('Peaks are the following: ');
fprintf('%.4e ', peaks);
fprintf('\n');

fq_peaks=fr(locs);
fprintf('Computed frequencies are the following: ');
fprintf('%d ', fq_peaks);
fprintf('\n');
```

Figure 14: Post-Processing script for FFT ctd.

After the FFT, the function "*findpeaks"* helps to create an array of *peaks* and *locs* based on a minimum height, and distance as evidenced in Figure 14. The minimum height represents the minimum amplitude of the Electric field on a specific resonance, while the minimum distance represents the minimum number of samples or the width between peaks. Since *df* decreases by a factor of 4 for every case, the opposite applies for the width, it increases by a factor of 4. For all cases, the 3 lowest modes, TM (1,1,0), TM (1,1,1), and TM (2,1,0) are identified from the FFT as seen in Figure 14.

Lastly, an additional “*RelativeError”* script is added outside the post processing block, to compute the relative error of the 1st and 3rd resonance in the simulation, “*fcomputed”,* versus the 1st and 3rd exact resonance calculated from equation (21), “*fanalytical*”. These values are stored for each case to create a convergence plot of the relative error vs the increase in *dx* as observed in Figure 15. The expectations are to demonstrate 2nd order accuracy, or in general words, a convergence plot with a slope of 2.



```
% In line script for convergence plot on relative error on 3 lowest modes
fcomputed=[fq_peaks(1),fq_peaks(3)];
fanalytical= [fex(1),fex(3)];

%Relative Error for TM(1,1,0), and TM(2,1,0)
RelError1= abs((fcomputed(1)-fanalytical(1))/fanalytical(1));
RelError2= abs((fcomputed(2)-fanalytical(2))/fanalytical(2));

%Dx values are stored from every case
dx_values=[0.2,0.1,0.05];

%Relative error stored from every case
TotalError1=[0.0063,0.0010,2.7977e-04];
TotalError2=[0.0296,0.0072,0.0018];

figure;
loglog(dx_values,TotalError1,'LineWidth',2);
hold on ; grid on
loglog(dx_values,TotalError2,'LineWidth',2)
ylabel('Relative Error')
xlabel('\Delta X ')
title('Error convergence Plot')
legend('TM(1,1,0)', 'TM(2,1,0)','Location','northwest');
```

Figure 15: Relative Error script.

4. CASE STUDIES VALIDATION

As mentioned in the input program control block, case study 1 has a number of cells, *nx, ny, nz* equal to 6, a sampling frequency, *fft_size,* equal to 2^{12}, a *width* of 20, a *height* of 3.5e-9, a source injection between nodes (2,2,2) and (2,2,3), and an output probe between nodes (4,4,4) and (4,4,5). It is observed from Figure 16 that the *df* value equals to 6.4026e+05, and that the 3 lowest computed resonant frequencies are 2.106455e+08, 2.593052e+08, and 3.252519e+08.

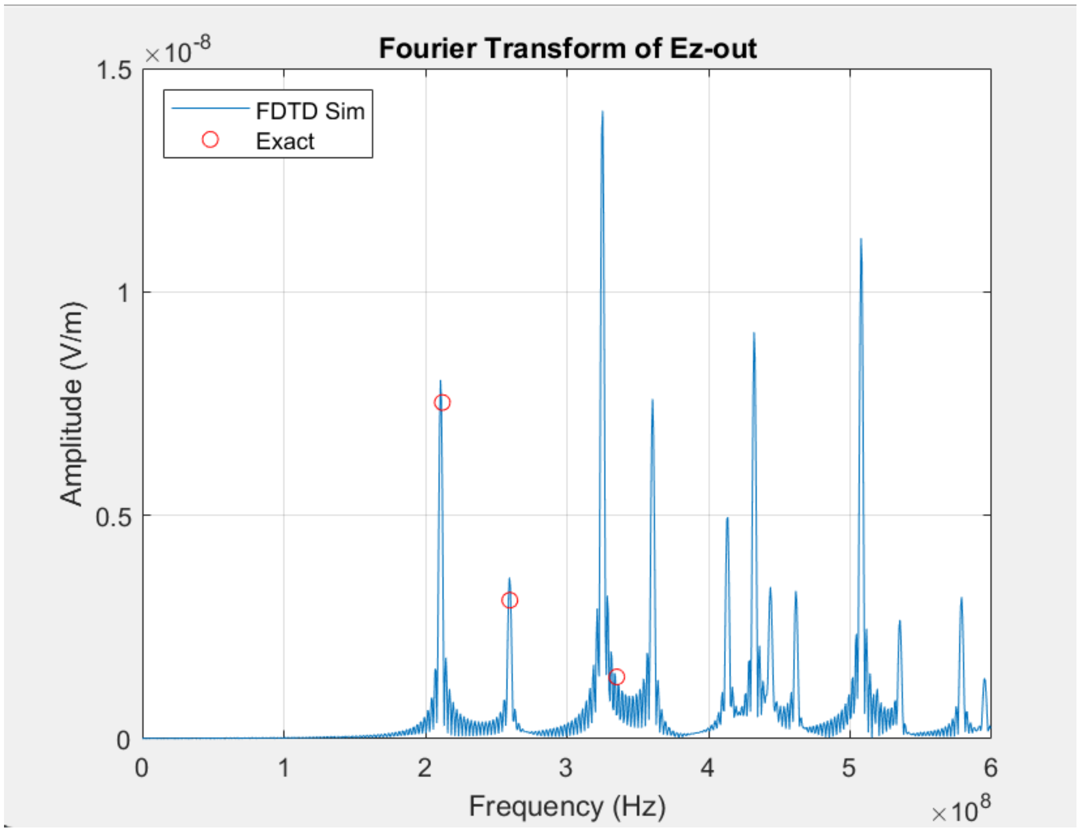


Figure 16: Case study 1 results

Case study 2 has a number of cells, *nx, ny, nz* equal to 11, a sampling frequency, *fft_size,* equal to 2^{15}, a *width* of 80, a *height* of 1.5e-9, a source injection between nodes (4,4,4) and (4,4,6), and an output probe between nodes (8,8,8) and (8,8,9). It is observed from Figure 17 that the *df* value equals 1.6006e+05, and that the 3 lowest computed resonant frequencies are 2.117659e+08, 2.593052e+08, and 3.327750e+08.

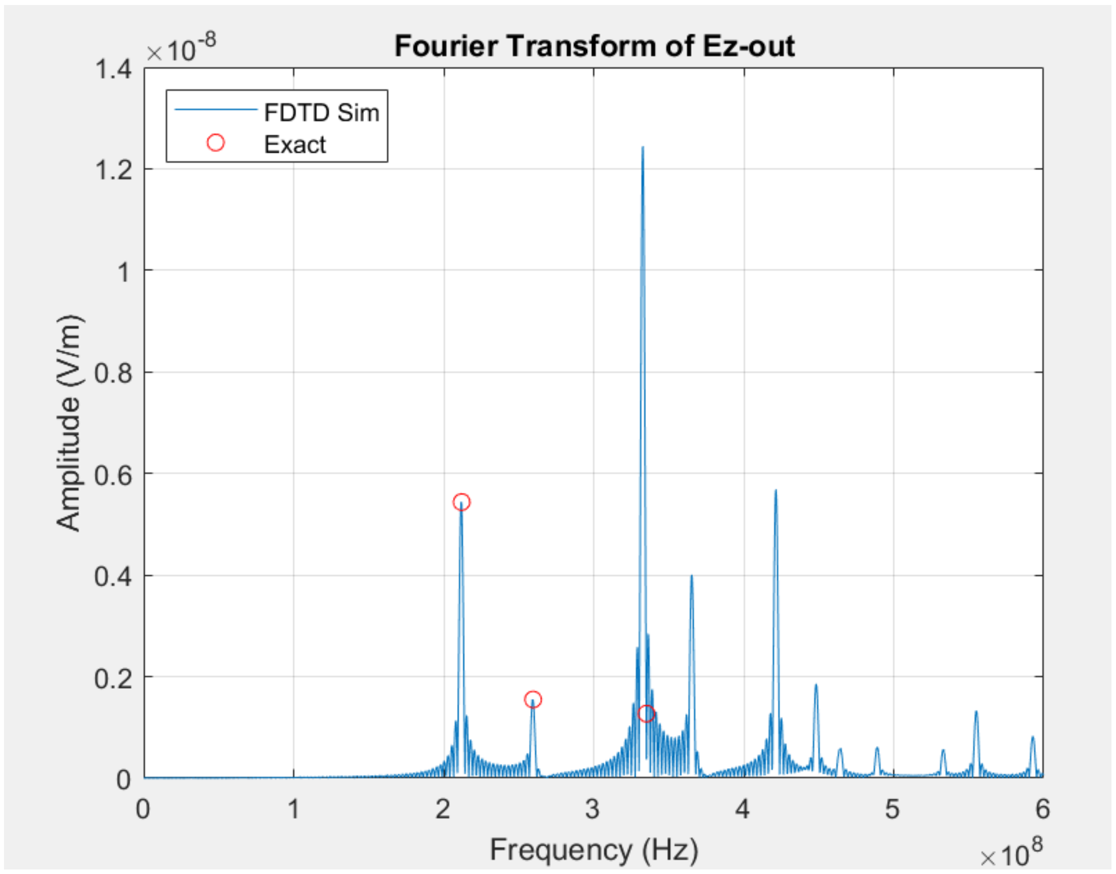


Figure 17: Case study 2 results

Case study 3 has a number of cells, *nx, ny, nz* equal to 21, a sampling frequency, *fft_size,* equal to 2^{18}, a *width* of 320, a *height* of 3.9e-10, a source injection between nodes (8,8,8) and (8,8,12), and an output probe between nodes (16,16,16) and (16,16,17). It is observed from Figure 18 that the *df* value equals 4.0016e+04, and that the 3 lowest computed resonant frequencies are 2.119260e+08, 2.594252e+08, and 3.345757e+08

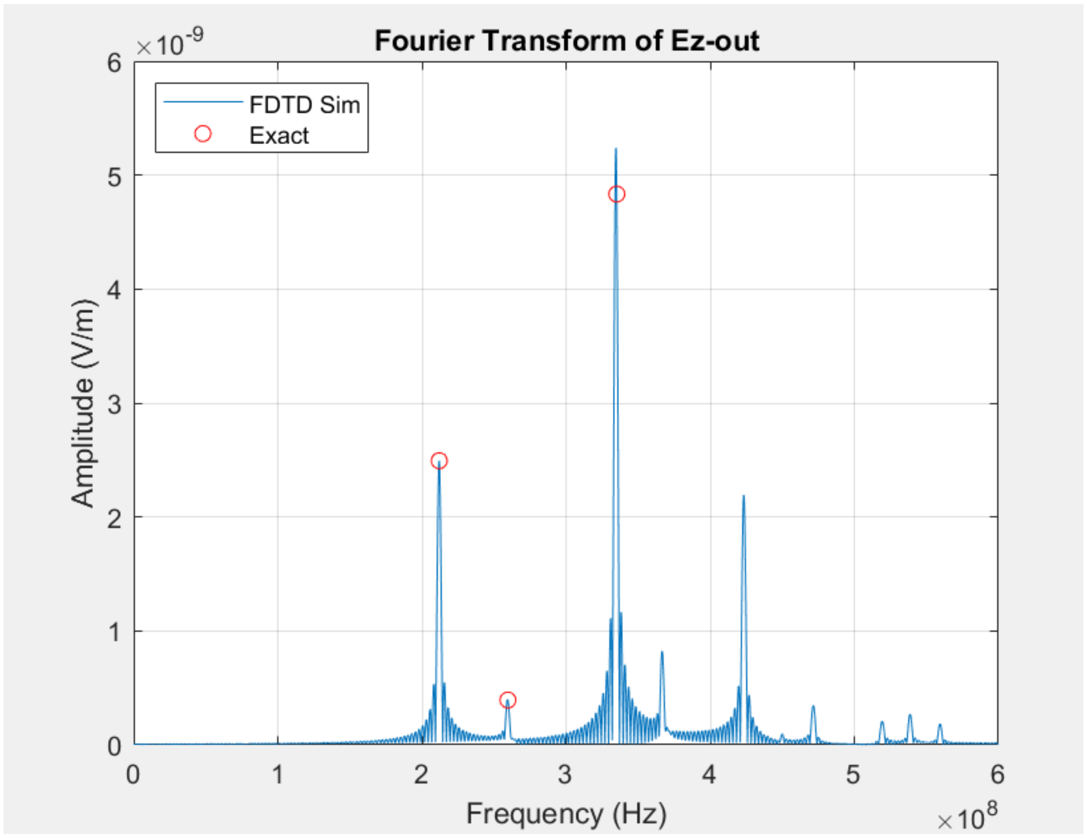


Figure 18: Case study 3 results

Lastly, once all cases have run the simulation, an error convergence *"loglog"* plot determines the order of accuracy of the results as observed in Figure 19. As mentioned previously the expectations are to demonstrate a 2nd order accuracy, or a slope of 2 in all curves. It is worth mentioning that the error convergence plot is done based on the 1st and 3rd resonant frequencies, TM (1,1,0), and TM (2,1,0) respectively. The relative error is calculated from the following equation:

$$Relative\ Error = \frac{|fcomputed - fanalytical|}{fanalytical} \tag{22}$$

It can be stated then that all cases are correctly validated based on the 2nd order accuracy of the FDTD method. Expectations are met as demonstrated in Figure 19.

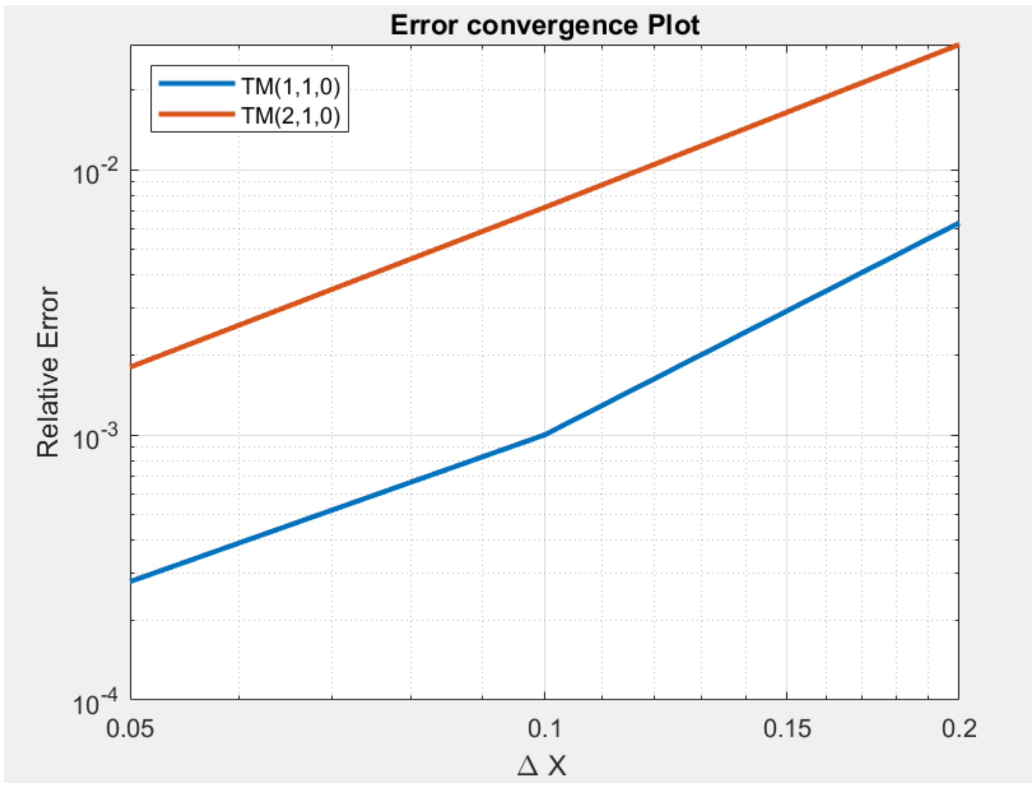


Figure 19: Error Convergence plot with a slope of 2.

5. SUMMARY

In conclusion, it can be said that all cases have met the expected results. The three-dimensional FDTD method has been successfully applied for the propagation of a differentiated gaussian pulse along the PEC Cavity. This is proven by the 2nd order accuracy of the error convergence plot, and by the decrease of *df* by a factor of 4 with each case. I consider this project a big step to understand how electromagnetic fields are propagated along space and time. In terms of MATLAB and programing in general, I believe my progress is undeniable. As an engineer and a student my objectives always are to expand my knowledge and also strive for excellence; for which I think this project helped me to get me closer to these goals. All cases validations are correct, concluding that the wave behavior will be influenced by the source configuration, space discretization, material properties, reflections, as well as the time step.

6. REFERENCES

SciPy Community. (n.d.). scipy.integrate.solve_ivp. *SciPy v1.10.0 Manual*. Retrieved 2/27/2025 from https://docs.scipy.org/doc/scipy/reference/generated/scipy.integrate.solve_ivp.html

MathWorks. (n.d.). *MATLAB documentation (Version R2024a)*. Retrieved 2/25/2025, from https://www.mathworks.com/help

MATLAB Crash Course for Beginners (2022, November 30). *Fundaments of MATLAB*. YouTube. Retrieved 2/22/2025 from https://www.youtube.com/watch?v=7f50sQYjNRA

Gedney, S.D. (2010). *Introduction to the Finite-Difference Time-Domain (FDTD) Method for Electromagnetics.* Springer Nature Switzerland AG. 10.1007/978-3-031-01712-4